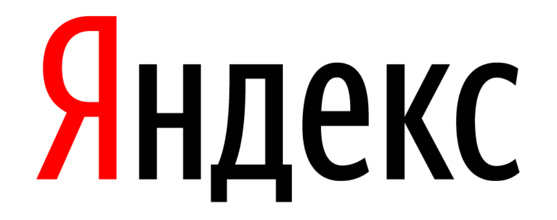




Надежные и быстрые бэкапы PostgreSQL

Даниил Захлыстов

В предыдущих сериях



Андрей Бородин, Георгий Рылов: Резервное копирование нагруженных СУБД > clck.ru/Ln8Qw

Андрей Бородин и Владимир Лесков: Масштабирование реплик PostgreSQL под нагрузкой с точки зрения технологий резервного копирования > clck.ru/F8ioz

Андрей Бородин: Разгоняем бэкап > clck.ru/Ebbte

Простой способ переноса данных

pg_dump



pg_dump

pg_dump — extract a PostgreSQL database into a script file or other archive file

postgresql.org/docs/current/static/app-pgdump.html

pg_dump



- Логическая копия ваших данных
- Не зависит от версии Pg
- Можете нарезать свою базу кусочками

Если всё серьёзно

Point in time recovery

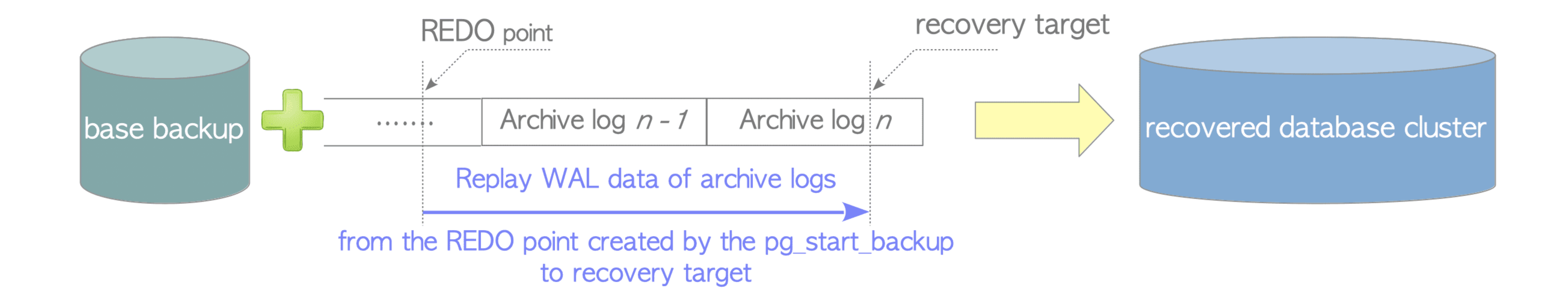


pg_start_backup API

- Checkpoint
- Full page writes = on
- Записывает LSN начала бэкапа

Не консистентная копия базы данных!

pg_start_backup API



http://www.interdb.jp/pg/img/fig-10-02.png

pg_basebackup

Использует pg_start_backup API, простой инструмент создания физической резервной копии

pg_basebackup

Использует pg_start_backup API, простой инструмент создания физической резервной копии, но хочется:

- Сжатие
- Параллелизм
- Шифрование
- Троттлинг ресурсов
- Листинг и управление
- Верификация

WAL-G

Тоже использует pg_start_backup API для создания физической резервной копии, и умеет много всего:

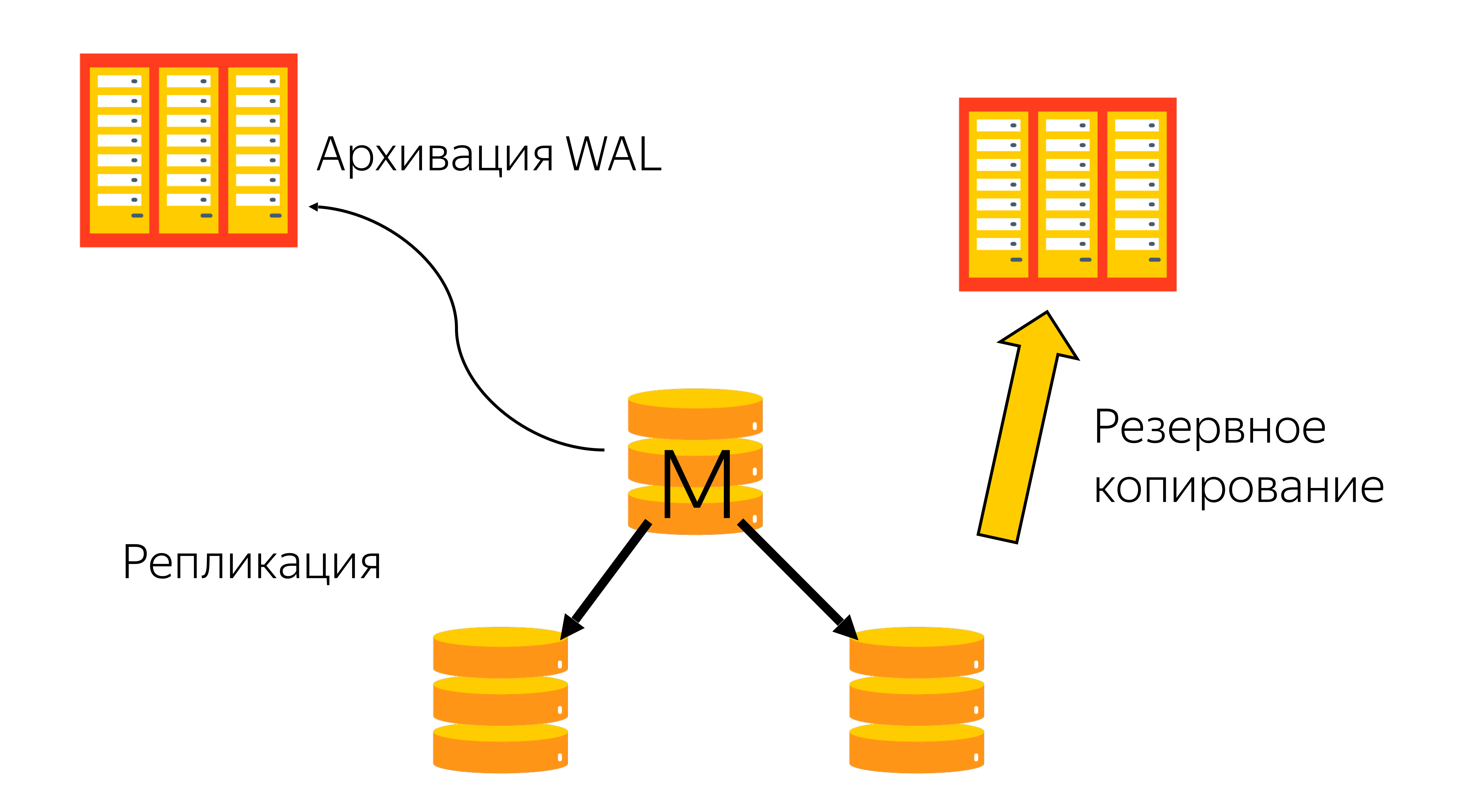
- Сжатие
- Параллелизм
- Шифрование
- Троттлинг ресурсов
- Листинг и управление
- Верификация

Что нового у WAL-G?

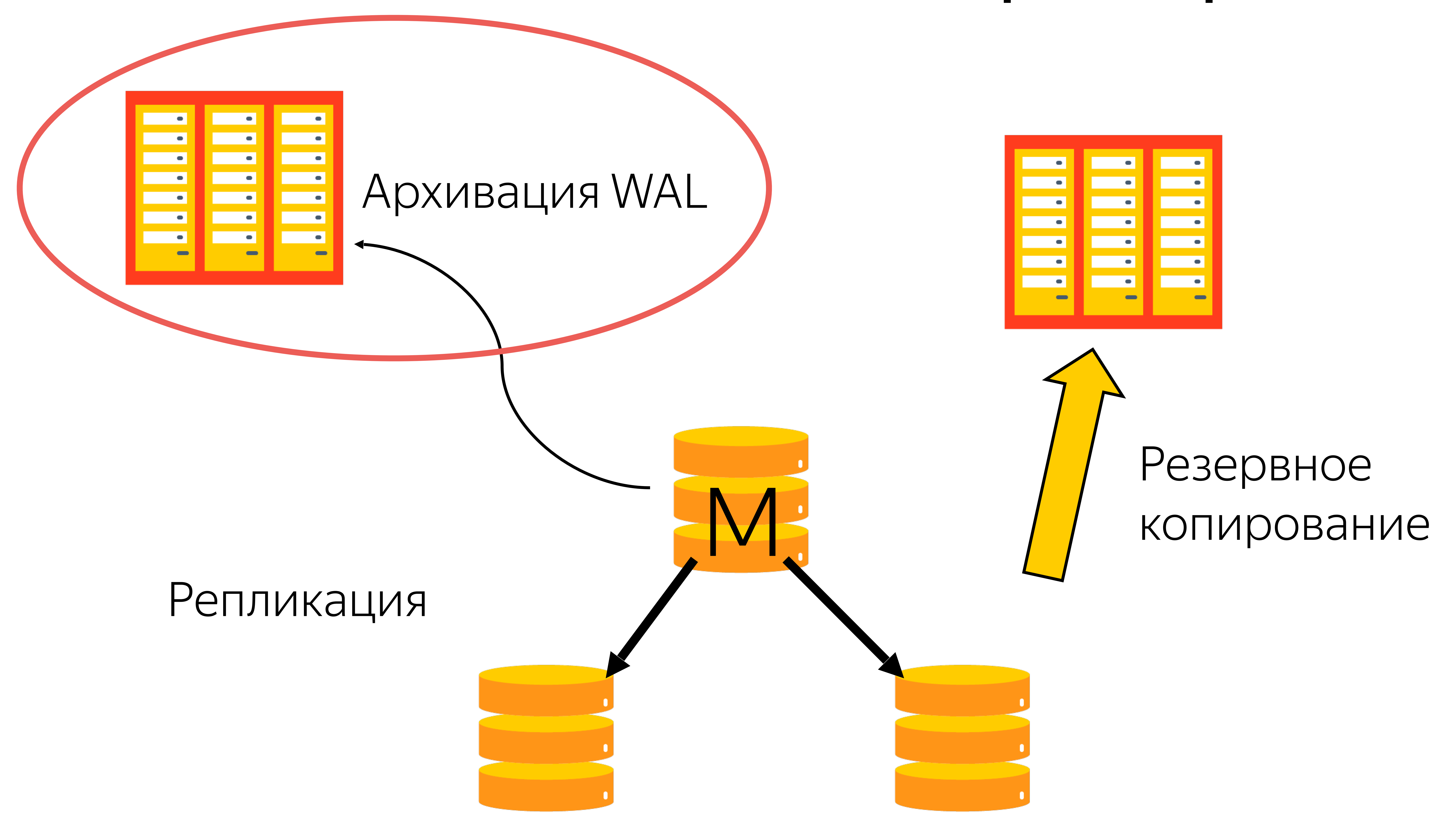


Верификация консистентности истории WAL

Топология HA-кластера с архивом



Топология HA-кластера с архивом



wal-g wal-push

```
# - Archiving -

archive_mode = on

archive_command = '/usr/bin/envdir /etc/wal-g/envdir
/usr/bin/timeout 600 /usr/bin/wal-g wal-push %p'
```

Архив WAL сегментов

000000010000001300000E1

000000010000001300000E2

000000010000001300000E3

000000010000001300000E4

000000010000001300000E5

http://wiki.postgresql.org/images/e/e5/FOSDEM2013-Timelines.pdf

Архив WAL сегментов

000000010000000013000000E1

000000010000000013000000E2

000000010000000013000000E3

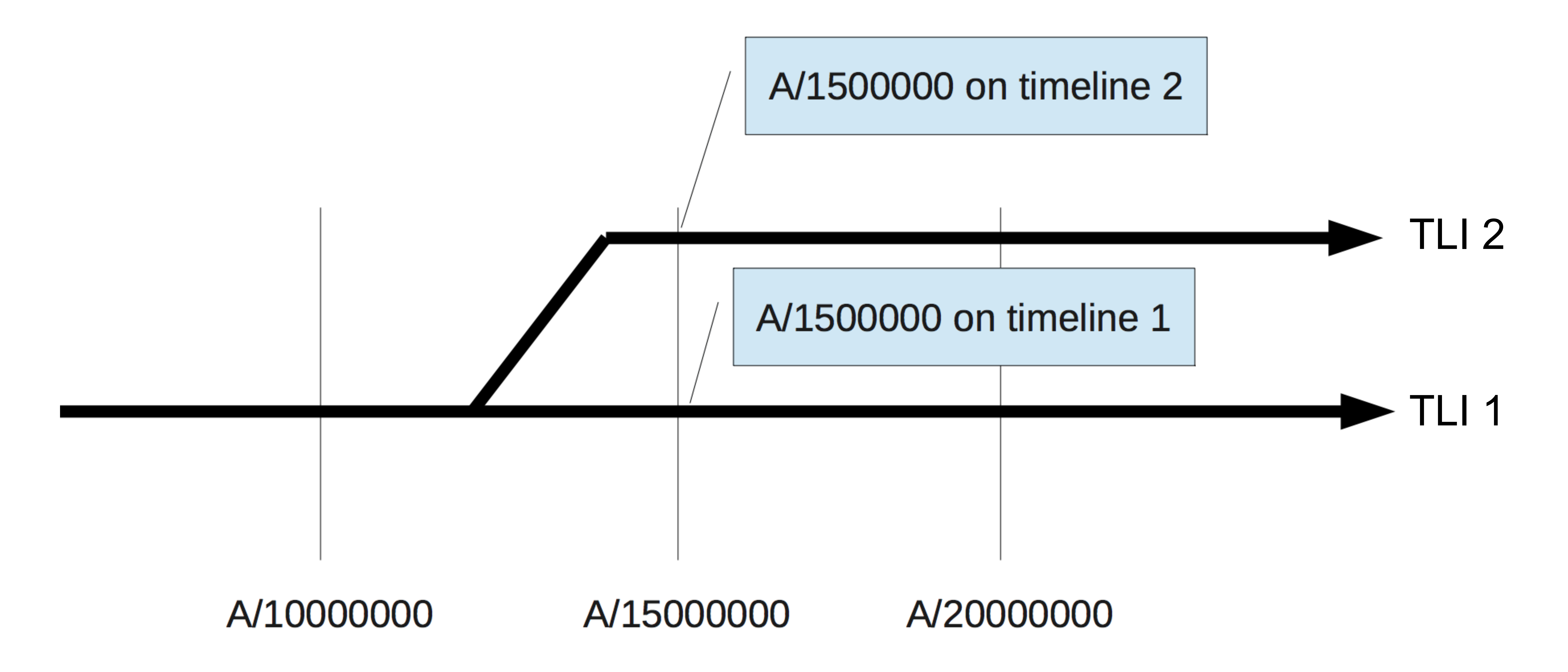
000000010000000013000000E4

000000010000000013000000E5

TLI LSN

http://wiki.postgresql.org/images/e/e5/FOSDEM2013-Timelines.pdf

PostgreSQL TimelineId



wiki.postgresql.org/images/e/e5/FOSDEM2013-Timelines.pdf

Архив WAL сегментов

000000010000001300000E1
000000010000001300000E2
000000010000001300000E3
000000010000001300000E4
000000010000001300000E5

http://wiki.postgresql.org/images/e/e5/FOSDEM2013-Timelines.pdf

Архив WAL сегментов

000000010000000013000000E1
000000010000000013000000E2
000000010000000013000000E3 000000020000000013000000E3
000000010000000013000000E4 000000020000000013000000E4
000000010000000013000000E5 000000020000000013000000E5

http://wiki.postgresql.org/images/e/e5/FOSDEM2013-Timelines.pdf

Архив WAL сегментов

000000010000001300000E1

000000010000001300000E2

000000010000001300000E3   000000020000001300000E3

000000010000001300000E4   000000020000001300000E4

000000010000001300000E5   000000020000001300000E5

\+ 00000002.history

http://wiki.postgresql.org/images/e/e5/FOSDEM2013-Timelines.pdf

Timeline .history file

```
root@some-host /var/lib/postgresql/13/data/pg_wal #
```

Timeline .history file

```
root@some-host /var/lib/postgresql/13/data/pg_wal # ls -l | grep .history
-rw------- 1 postgres postgres       50 Apr 23 15:24 00000002.history
-rw------- 1 postgres postgres       93 Apr 26 12:56 00000003.history
-rw------- 1 postgres postgres      136 Apr 27 09:25 00000004.history
root@some-host /var/lib/postgresql/13/data/pg_wal #
```

Timeline .history file

```
root@some-host /var/lib/postgresql/13/data/pg_wal # ls -l | grep .history
-rw------- 1 postgres postgres       50 Apr 23 15:24 00000002.history
-rw------- 1 postgres postgres       93 Apr 26 12:56 00000003.history
-rw------- 1 postgres postgres      136 Apr 27 09:25 00000004.history
root@some-host /var/lib/postgresql/13/data/pg_wal # cat 00000004.history
1 0/9007278 before 2021-04-23 15:01:39.230187+03

2 1/AE0000A0  no recovery target specified

3 2/28018600  no recovery target specified
```

Timeline .history file

```
root@some-host /var/lib/postgresql/13/data/pg_wal # ls -l | grep .history
-rw------- 1 postgres postgres       50 Apr 23 15:24 00000002.history
-rw------- 1 postgres postgres       93 Apr 26 12:56 00000003.history
-rw------- 1 postgres postgres      136 Apr 27 09:25 00000004.history
root@some-host /var/lib/postgresql/13/data/pg_wal # cat 00000004.history
1 0/9007278 before 2021-04-23 15:01:39.230187+03

2 1/AE0000A0  no recovery target specified

3 2/28018600  no recovery target specified
```

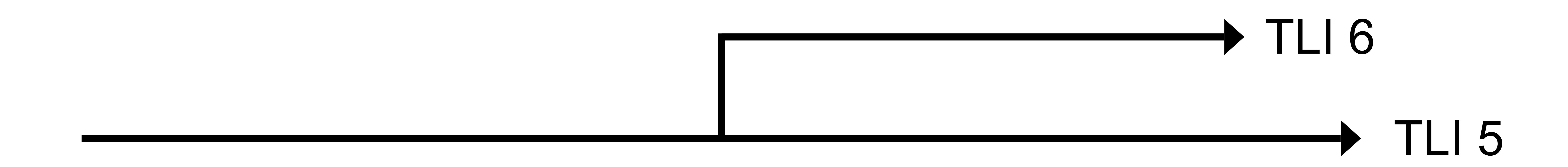
timeline_id   timeline_switch_lsn   comment

Проблемы



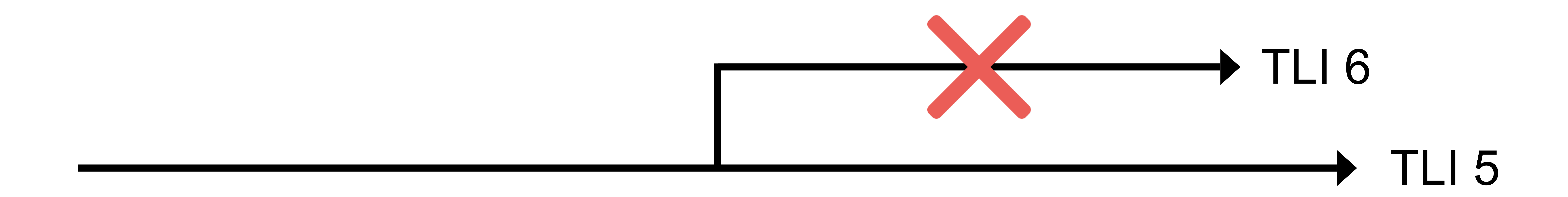
Split brain
Ситуация, в которой в HA-кластере появляется второй мастер. Происходят из-за проблем с сетевой связностью и / или проблем с сервисом координации. В качестве последствий - рассогласование данных в облачном хранилище WAL сегментов, так как возникает два конфликтующих таймлайна

Проблемы



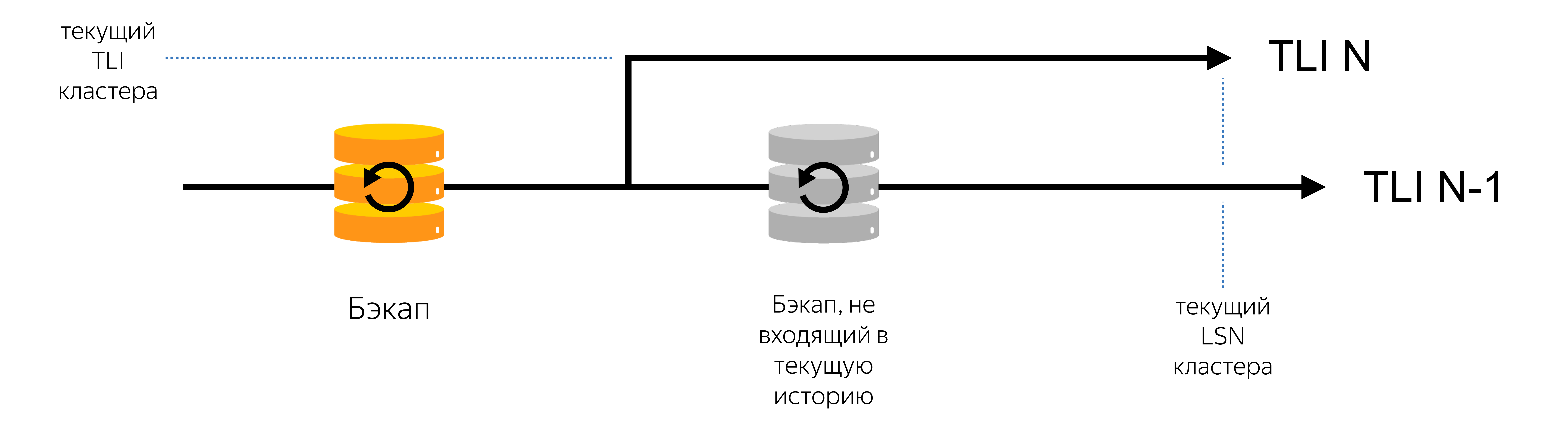
Split brain
Ситуация, в которой в HA-кластере появляется второй мастер. Происходят из-за проблем с сетевой связностью и / или проблем с сервисом координации. В качестве последствий - рассогласование данных в облачном хранилище WAL сегментов, так как возникает два конфликтующих таймлайна

Проблемы

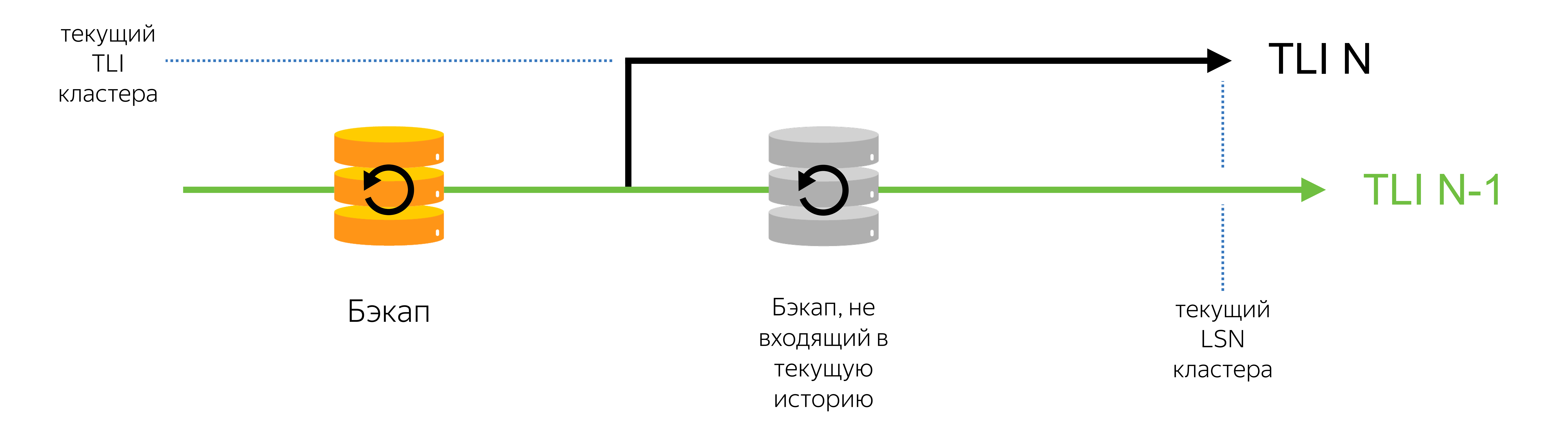


Split brain
Ситуация, в которой в HA-кластере появляется второй мастер. Происходят из-за проблем с сетевой связностью и / или проблем с сервисом координации. В качестве последствий - рассогласование данных в облачном хранилище WAL сегментов, так как возникает два конфликтующих таймлайна

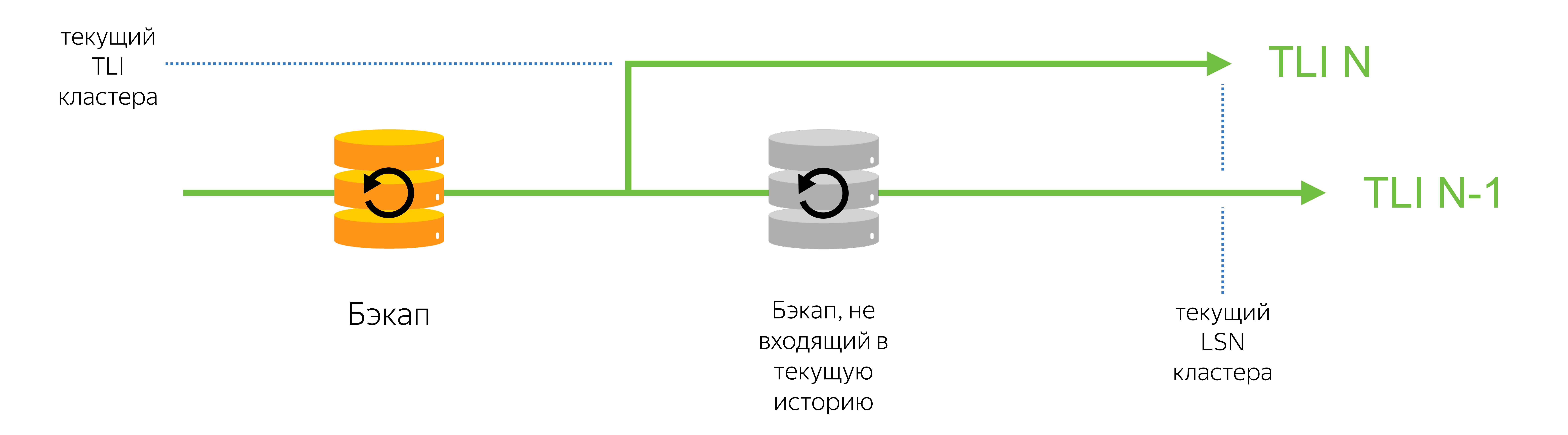
Проверка на неизвестные таймлайны



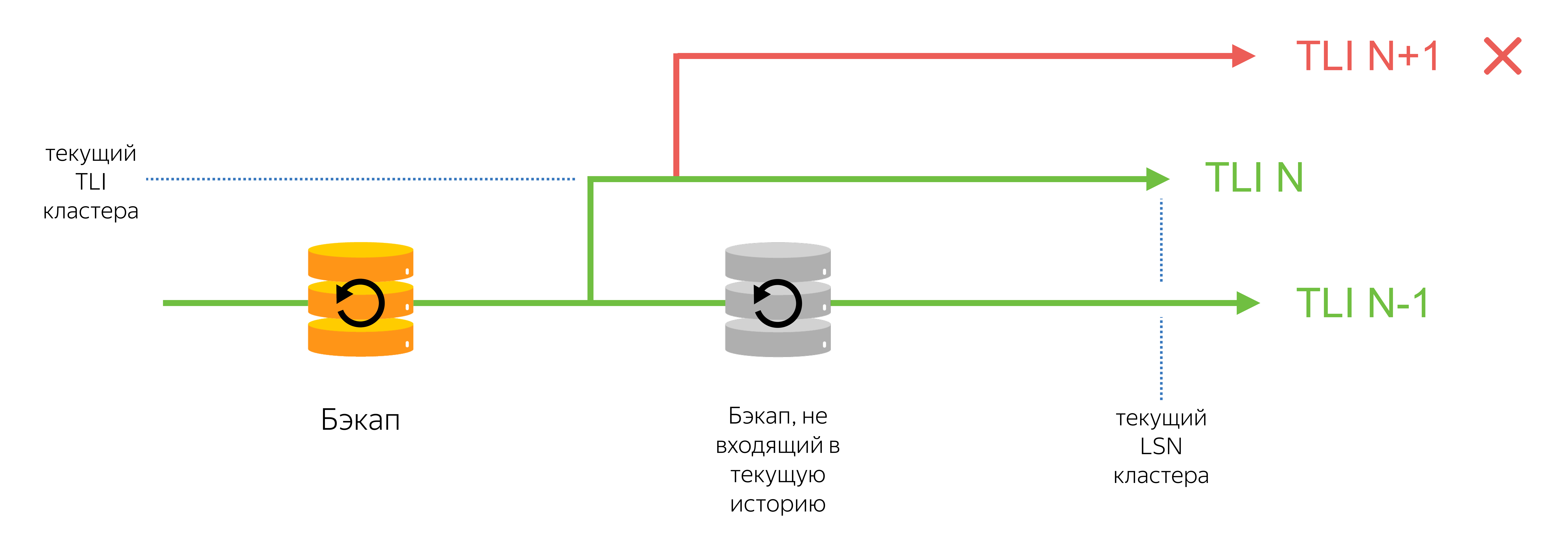
Проверка на неизвестные таймлайны



Проверка на неизвестные таймлайны



Проверка на неизвестные таймлайны



Проблемы

000000010000001З000000E1

~~000000010000001З000000E2~~

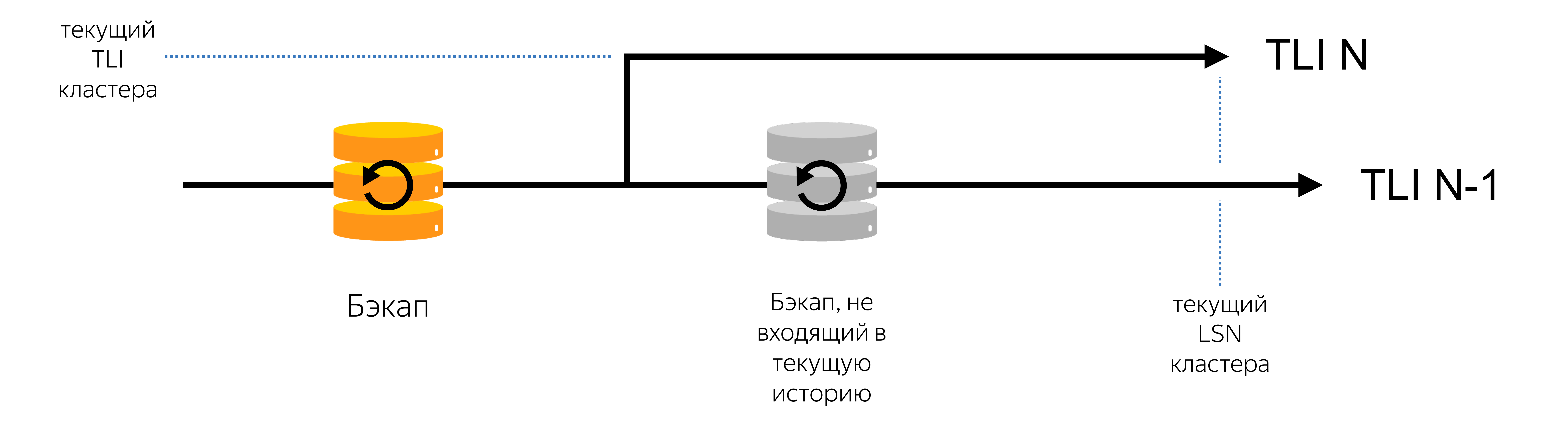
000000010000001З000000E3

000000010000001З000000E4

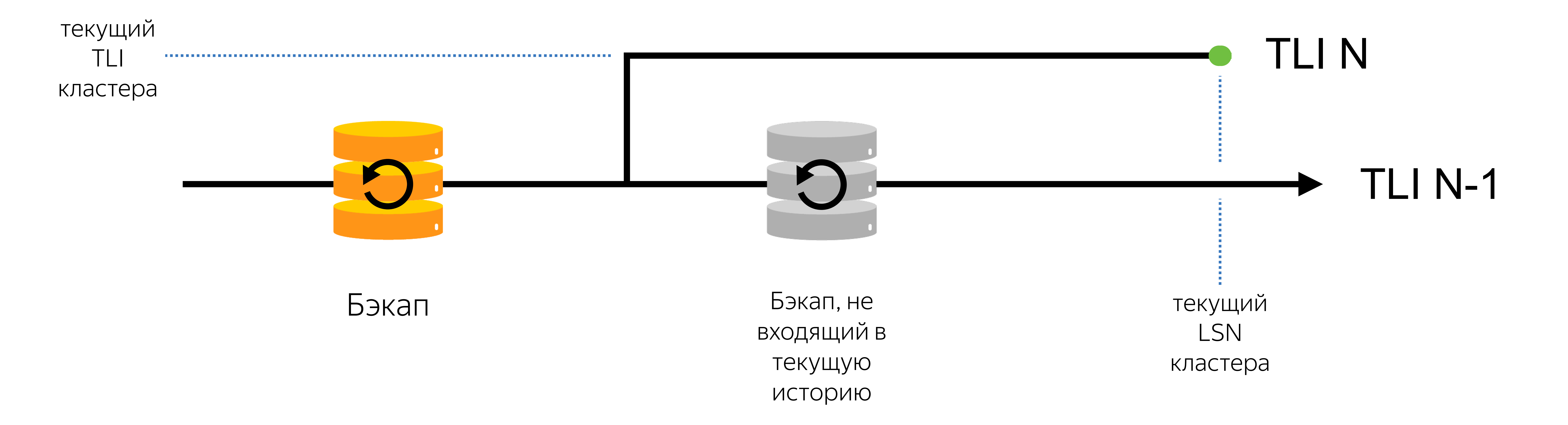
000000010000001З000000E5

Недоступность PITR
Потеря WAL сегмента или .history файла вызовет невозможность выполнить PITR

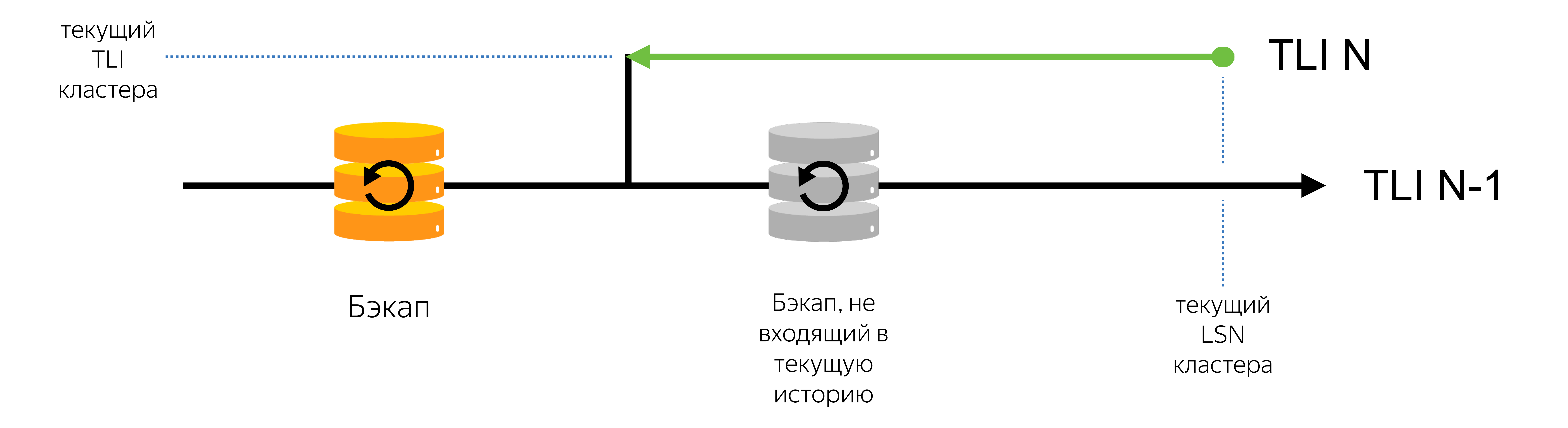
Проверка доступности PITR



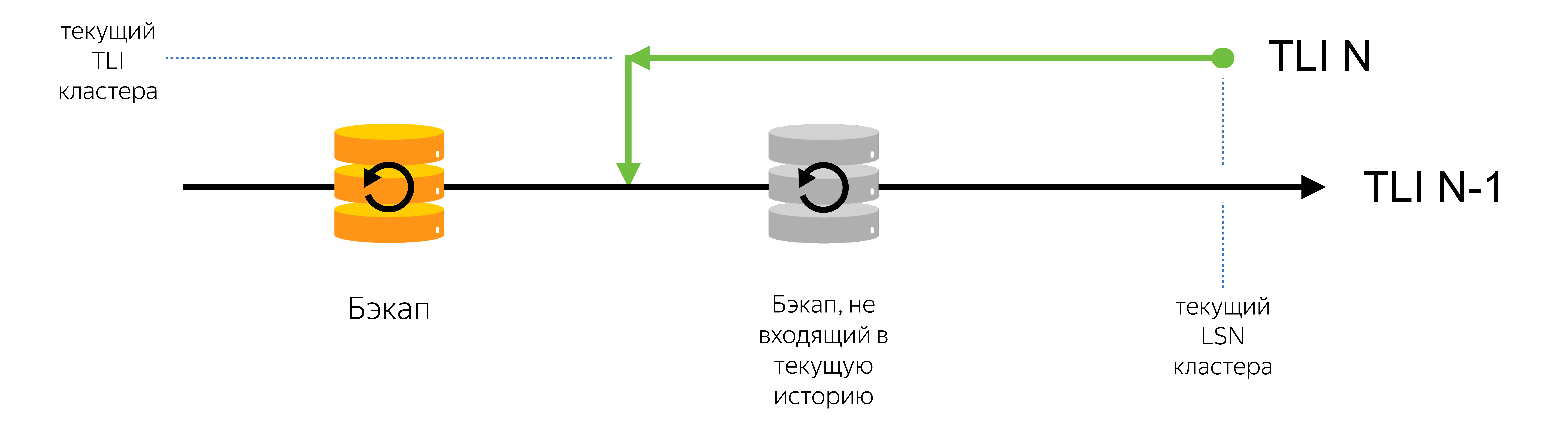
Проверка доступности PITR



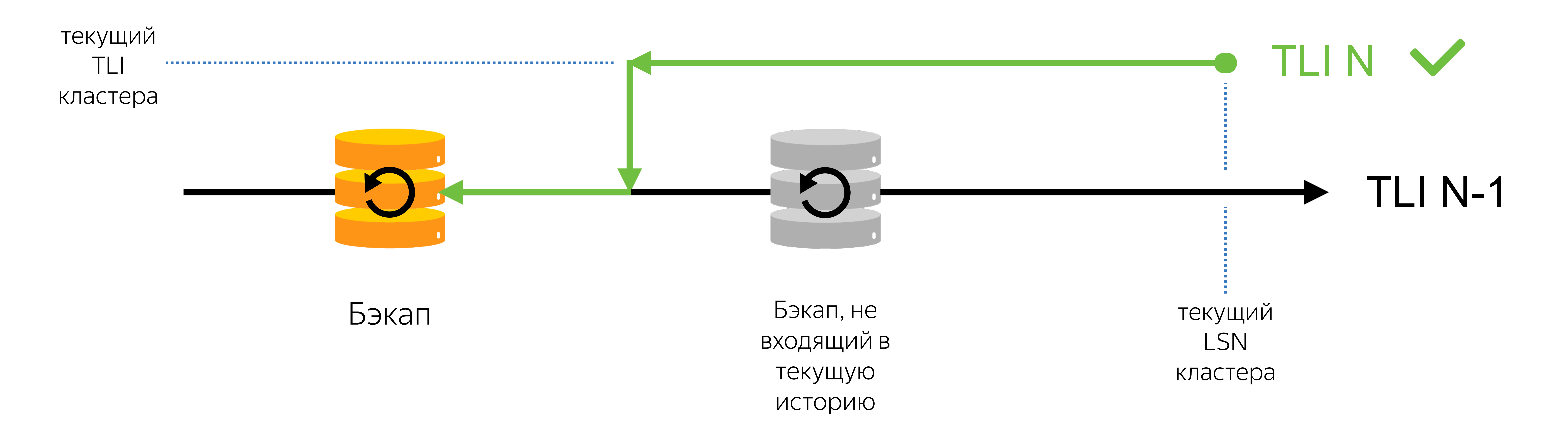
Проверка доступности PITR



Проверка доступности PITR



Проверка доступности PITR



Как использовать?

wal-verify — новая команда в WAL-G, выполняющая проверки состояния архива WAL в облачном хранилище

Проверка на неизвестные таймлайны

- wal-g wal-verify timeline

```
root@some-machine ~ # wal-g wal-verify timeline --config /etc/wal-g/wal-g.yaml
INFO: 2021/05/05 16:09:13.289940 Building check runner: timeline
INFO: 2021/05/05 16:09:13.289979 Running the check: timeline
[wal-verify] timeline check status: OK
[wal-verify] timeline check details:
Highest timeline found in storage: 4
Current cluster timeline: 4
```

Проверка доступности PITR

- wal-g wal-verify integrity

```
root@some-machine ~ # wal-g wal-verify integrity --config /etc/wal-g/wal-g.yaml
INFO: 2021/05/05 16:05:59.459085 Building check runner: integrity
INFO: 2021/05/05 16:05:59.496998 Detected earliest available backup: base_00000002000000000000000A
INFO: 2021/05/05 16:05:59.497033 Running the check: integrity
[wal-verify] integrity check status: OK
[wal-verify] integrity check details:
+-----+--------------------------+--------------------------+----------------+--------+
| TLI | START                    | END                      | SEGMENTS COUNT | STATUS |
+-----+--------------------------+--------------------------+----------------+--------+
|   2 | 00000002000000000000000A | 0000000200000001000000AD |            420 |  FOUND |
|   3 | 0000000300000001000000AE | 000000030000000200000027 |            122 |  FOUND |
|   4 | 000000040000000200000028 | 0000000400000006000000CF |           1192 |  FOUND |
+-----+--------------------------+--------------------------+----------------+--------+
```


когда используешь проверку
архива WAL сегментов

Верификация чексумм страниц

Проблемы

```
ERROR:  could not read block 274179 in file "base/13642/24643.2": read only 0 of 8192 bytes

ERROR:  could not access status of transaction 3250922107
DETAIL:  Could not open file "pg_xact/0C1C": No such file or directory.

ERROR: failed to re-find parent key in index "pg_attribute_relid_attnum_index" for split pages 6424/6425

PANIC could not locate a valid checkpoint record

ERROR:  found multixact 68834765 from before relminmxid 73262006

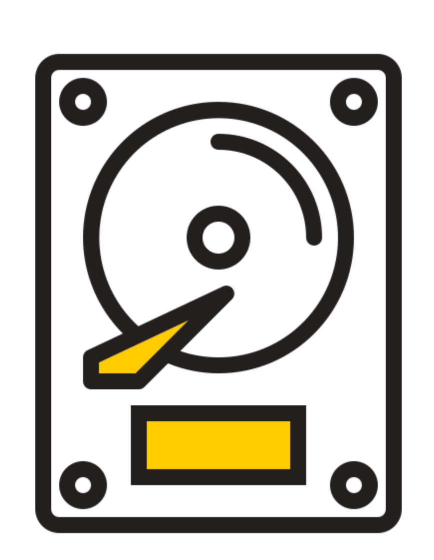
Error:  failed to re-find parent key in the index "xxx" for split pages yyy/zzz

ERROR:  cache lookup failed for type 16292881

PANIC could not locate a valid checkpoint record
```

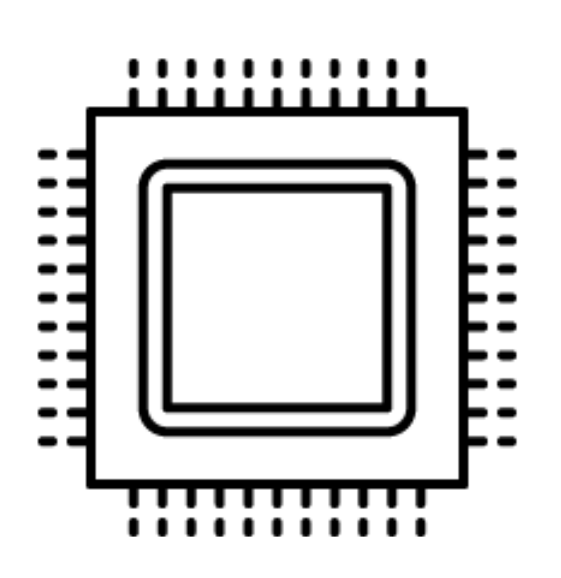
Коррупции данных

Причины коррупций

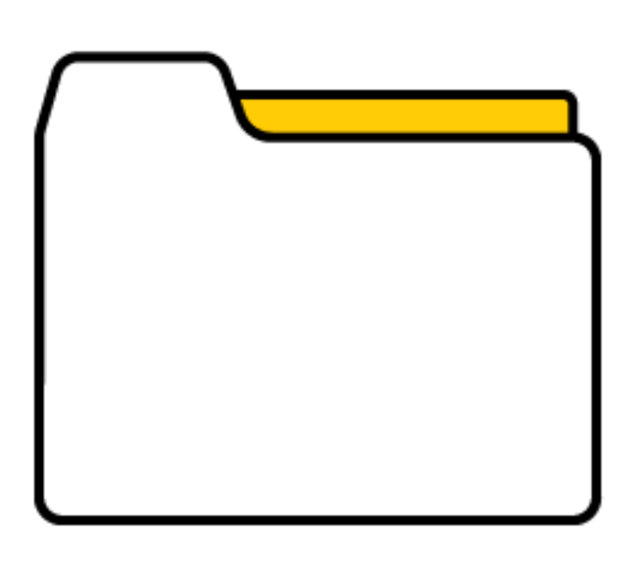




Аппаратные



Ошибки в микропрограммах



Файловая система

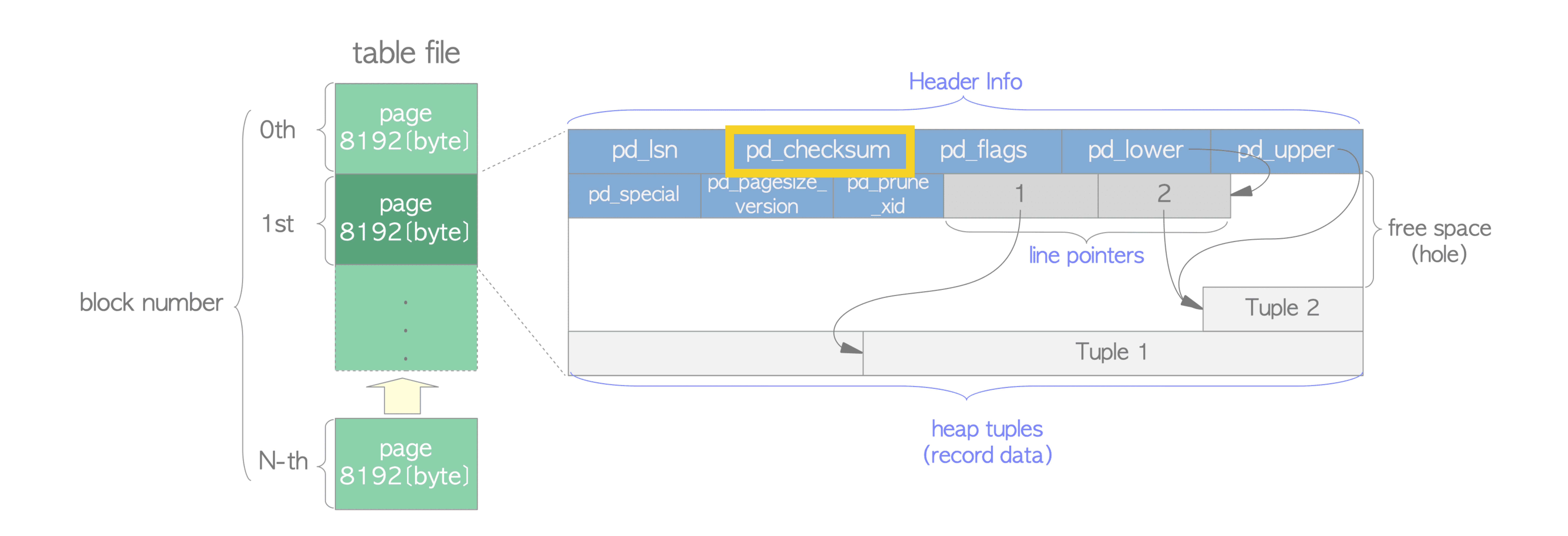


PostgreSQL



Ошибки бэкапа / репликации

Проверка чексумм в PostgreSQL



https://www.interdb.jp/pg/img/fig-1-04.png

Проверка чексумм в PostgreSQL

```
> SELECT * FROM some_table;

...

ERROR: invalid page in block 20 of relation base/19554/19584
```

https://bsdmag.org/page-checksum-protection-in-postgresql/

Проверка чексумм в WAL-G

- **Зачем?**
  Определить потенциально проблемный бэкап как можно раньше

- **Как включить?**
  С помощью флага:
  wal-g backup-push /path --verify

  В конфиге:
  WALG_VERIFY_PAGE_CHECKSUMS=TRUE

Проверка чексумм в WAL-G

***_backup_sentinel.json:**

```
...
"/base/16384/16397":{
   "CorruptBlocks":{
      "SomeCorruptBlocks":[3,5,17], // по умолчанию, до 10 первых поврежденных блоков
      "CorruptBlocksCount":3 // общее число поврежденных блоков
   },
   "IsIncremented":false,
   "IsSkipped":false,
   "MTime":"2020-08-27T14:31:06.483880188+05:00"
},
...
```

когда используешь проверку

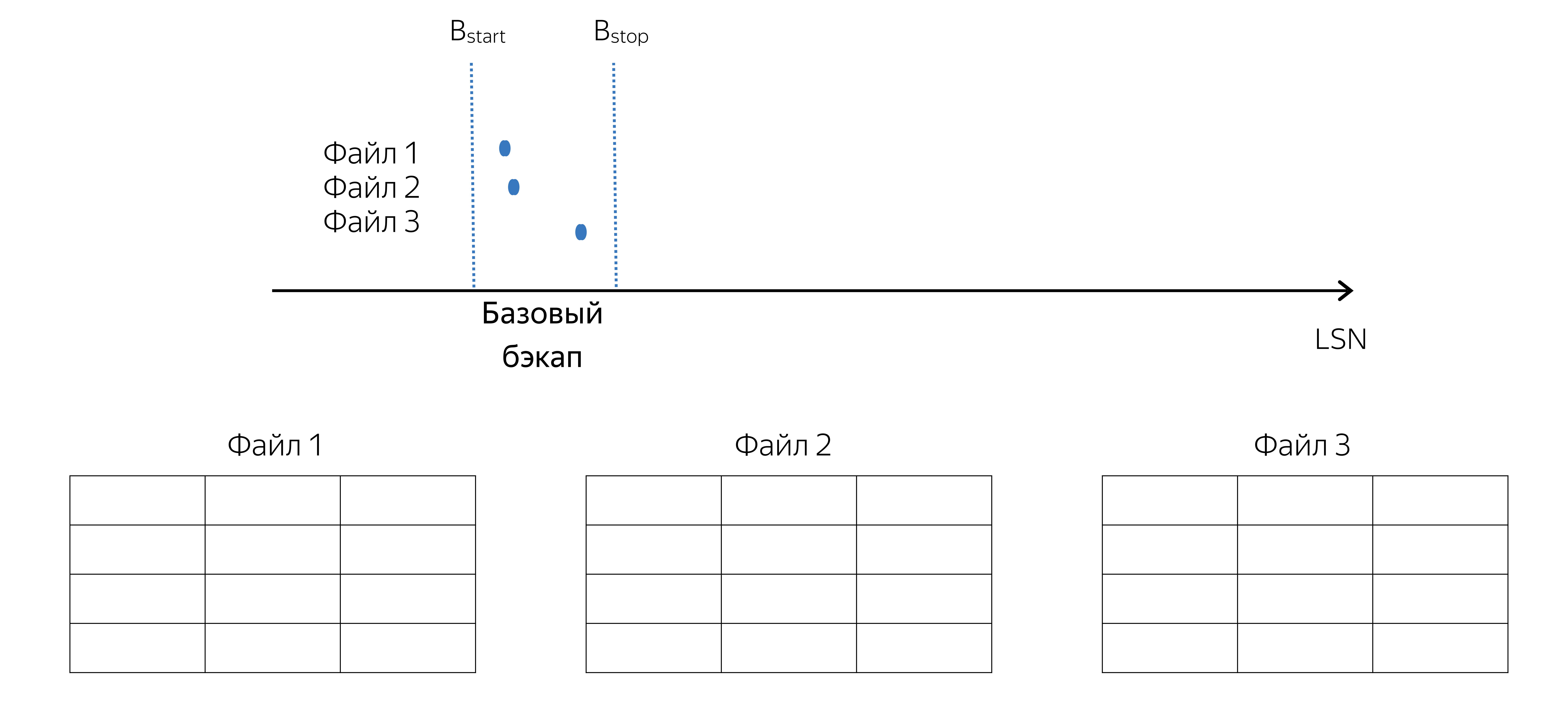
архива WAL сегментов

когда поставил

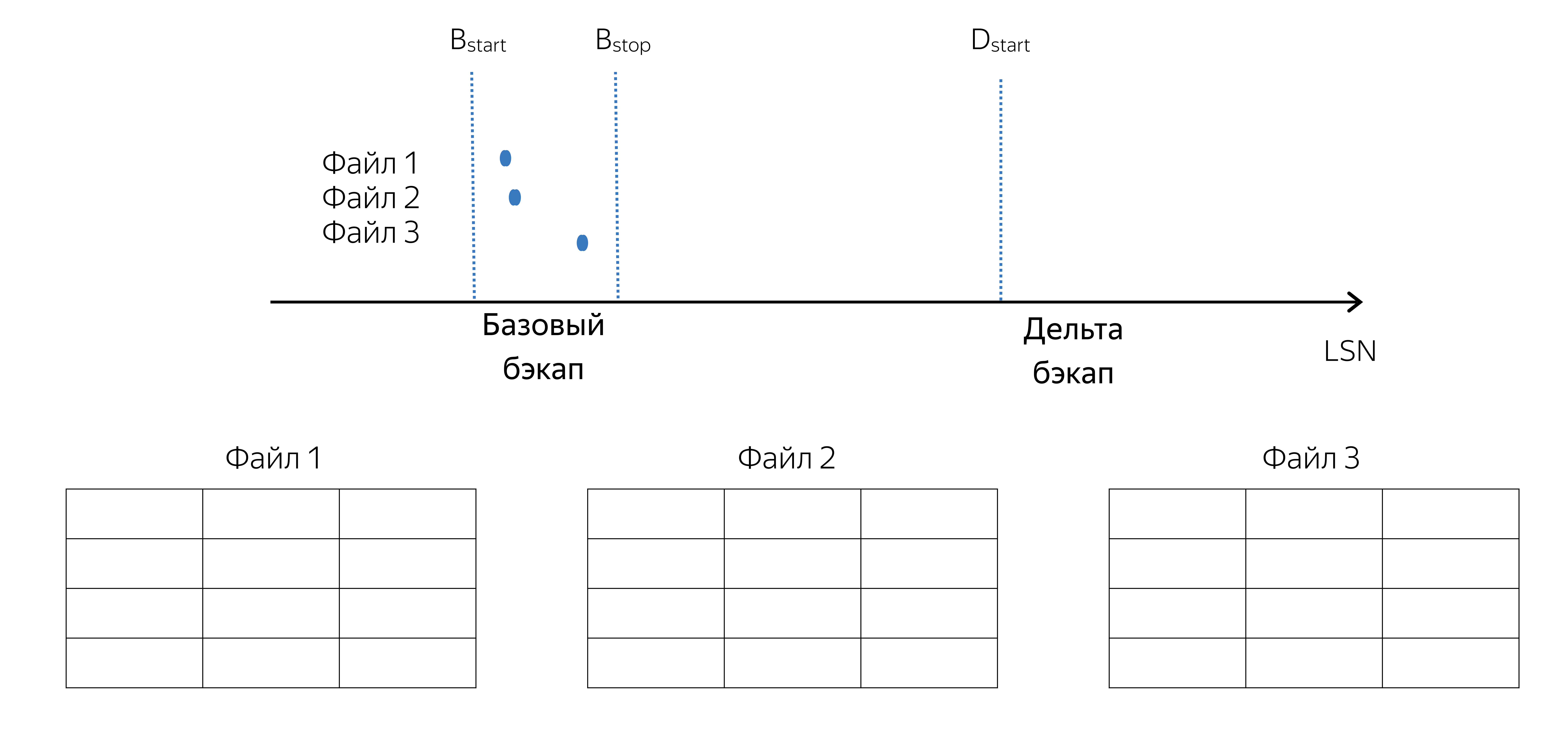
WALG_VERIFY_PAGE_CHECKSUMS
=TRUE

Обратная распаковка дельта бэкапов

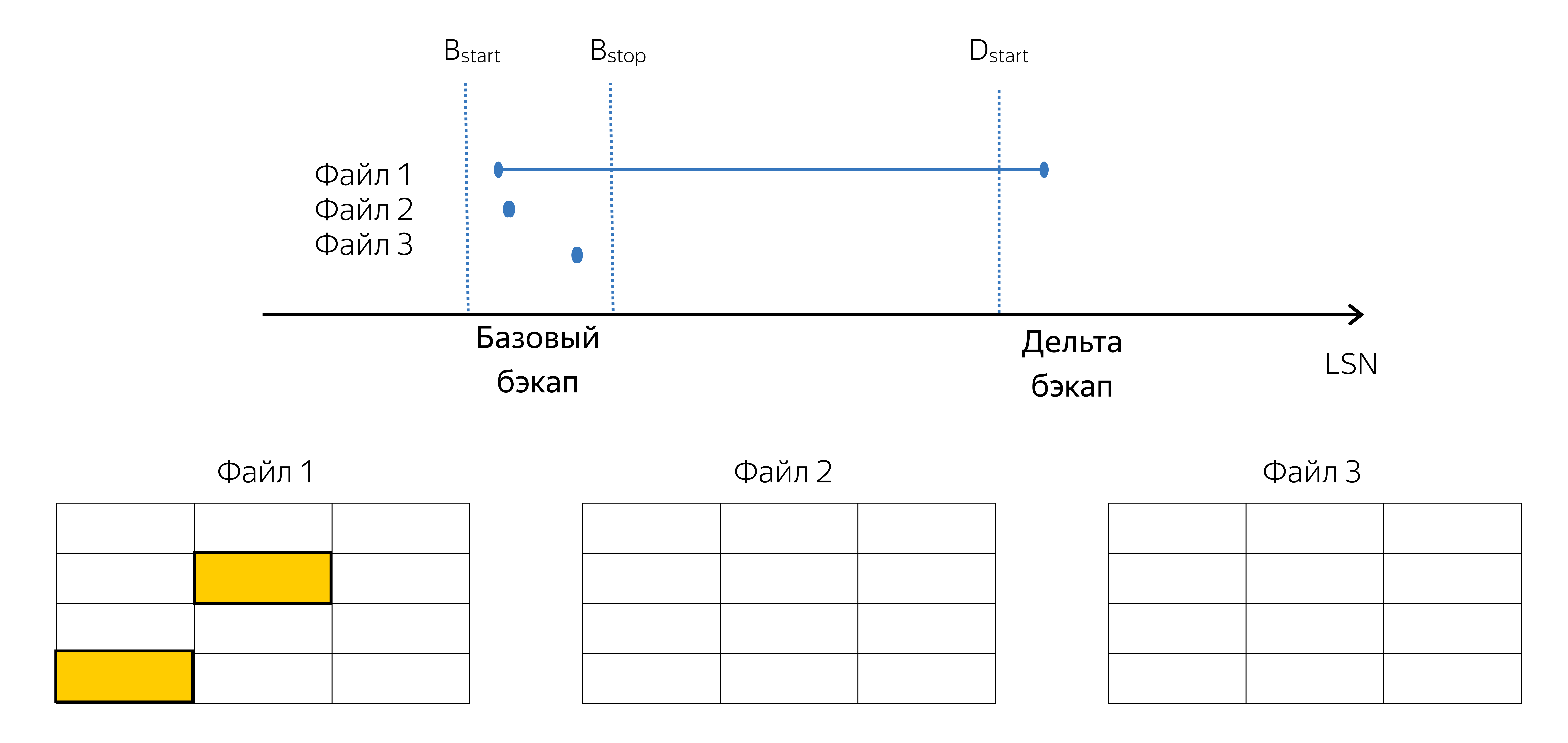
LSN-based дельты



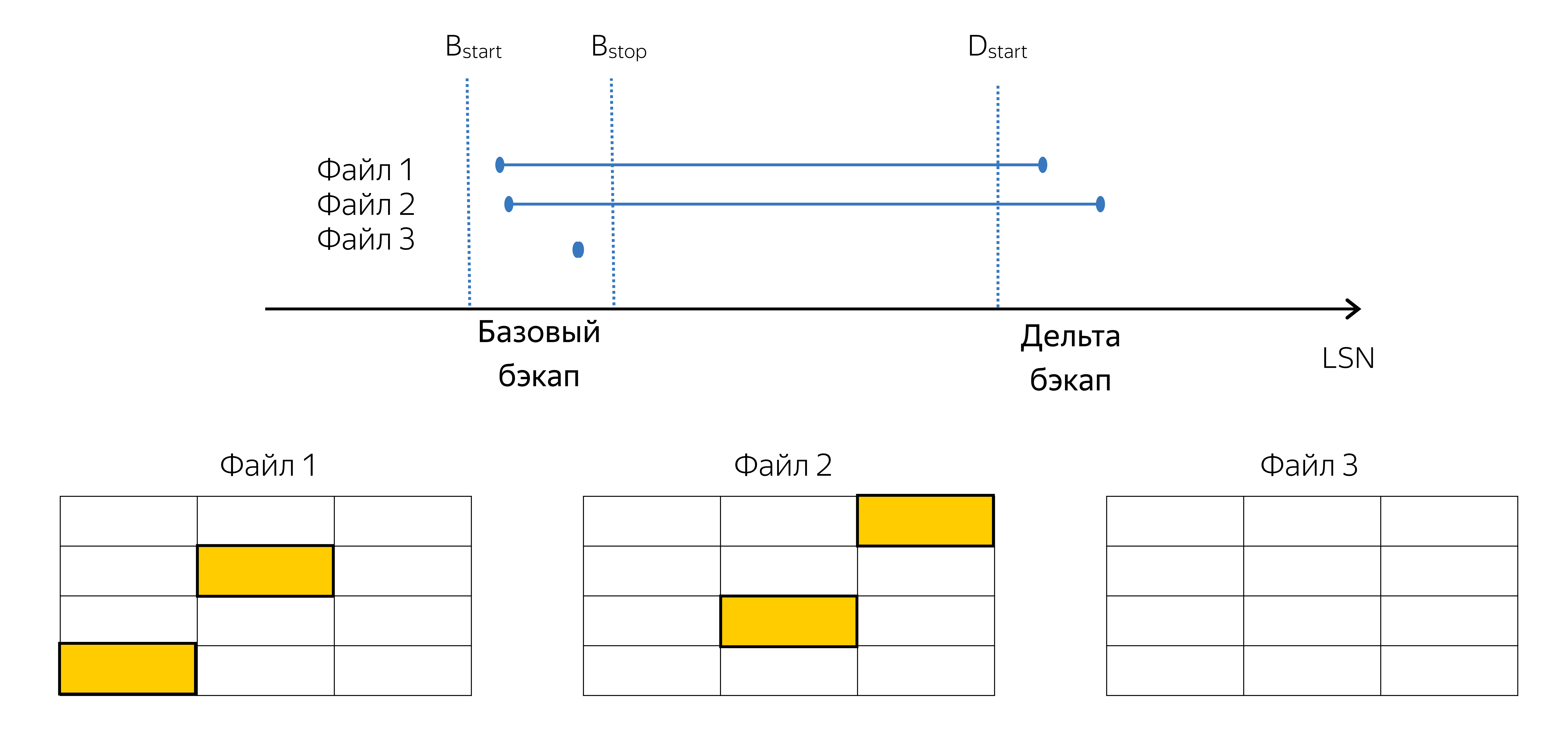
LSN-based дельты



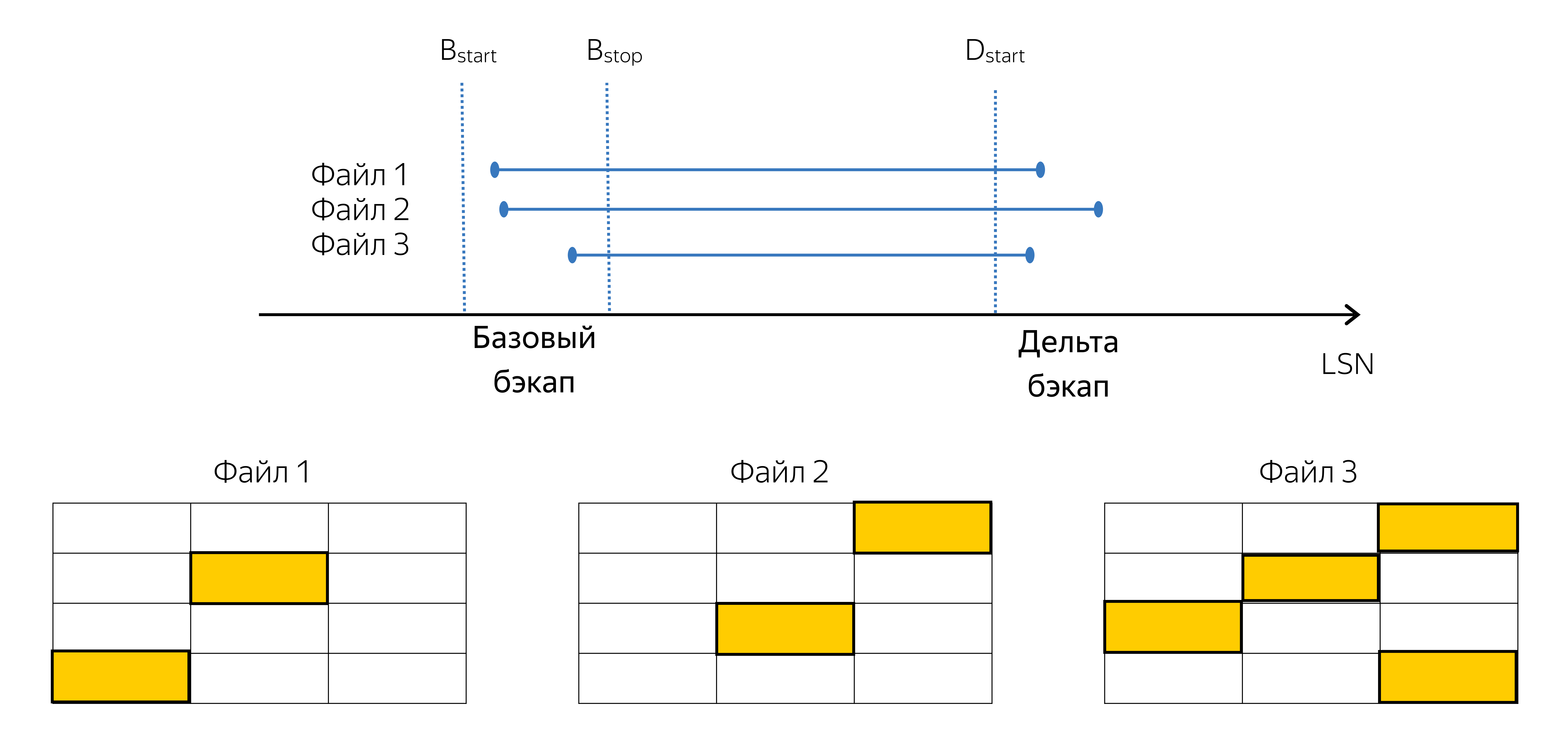
LSN-based дельты



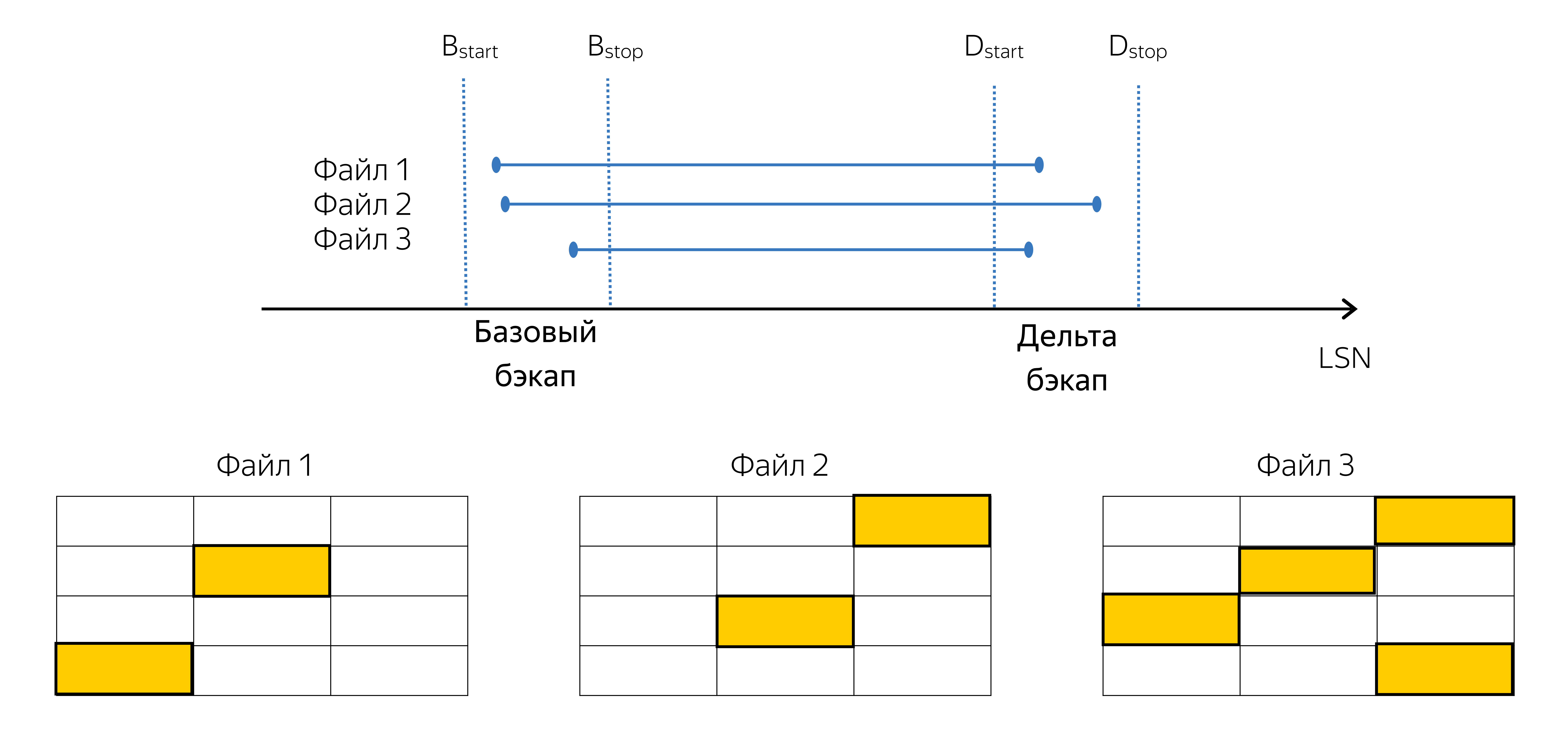
LSN-based дельты



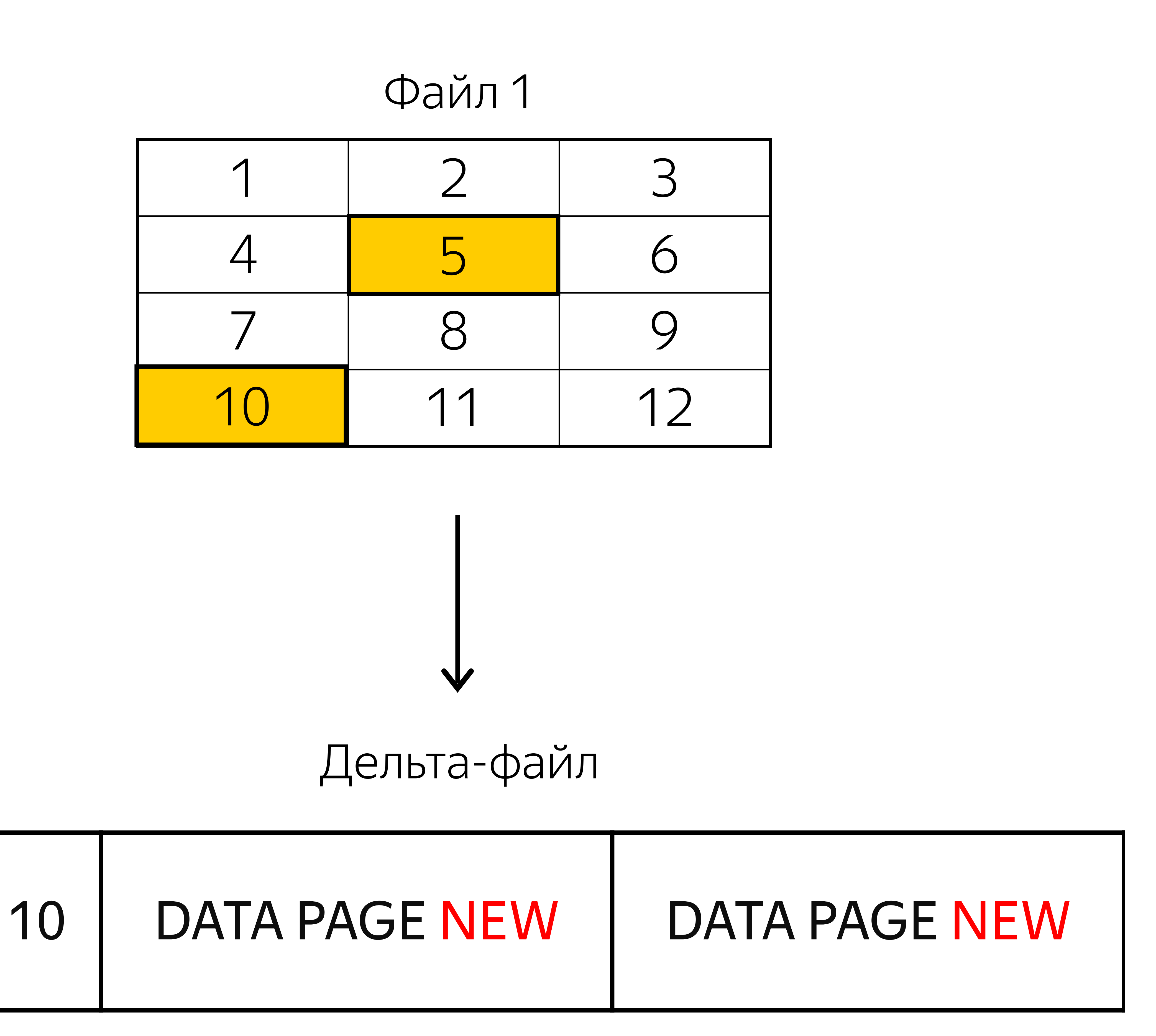
LSN-based дельты



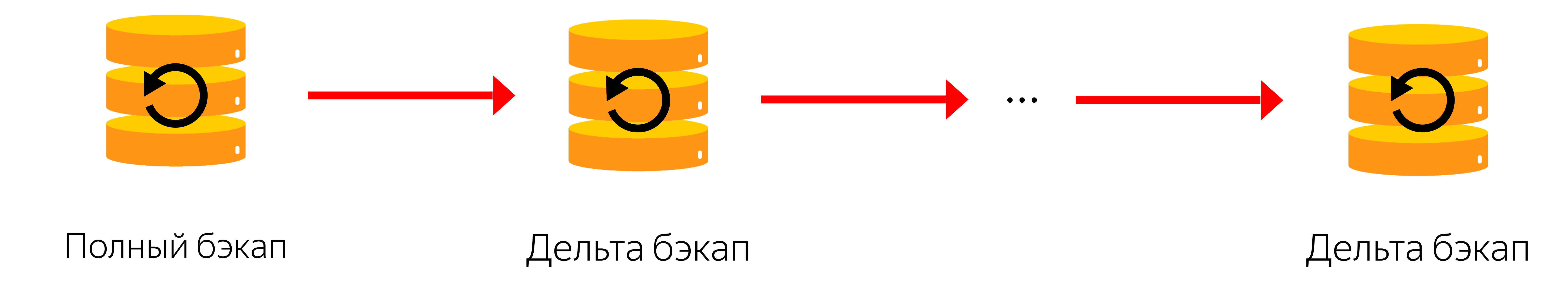
LSN-based дельты



LSN-based дельты

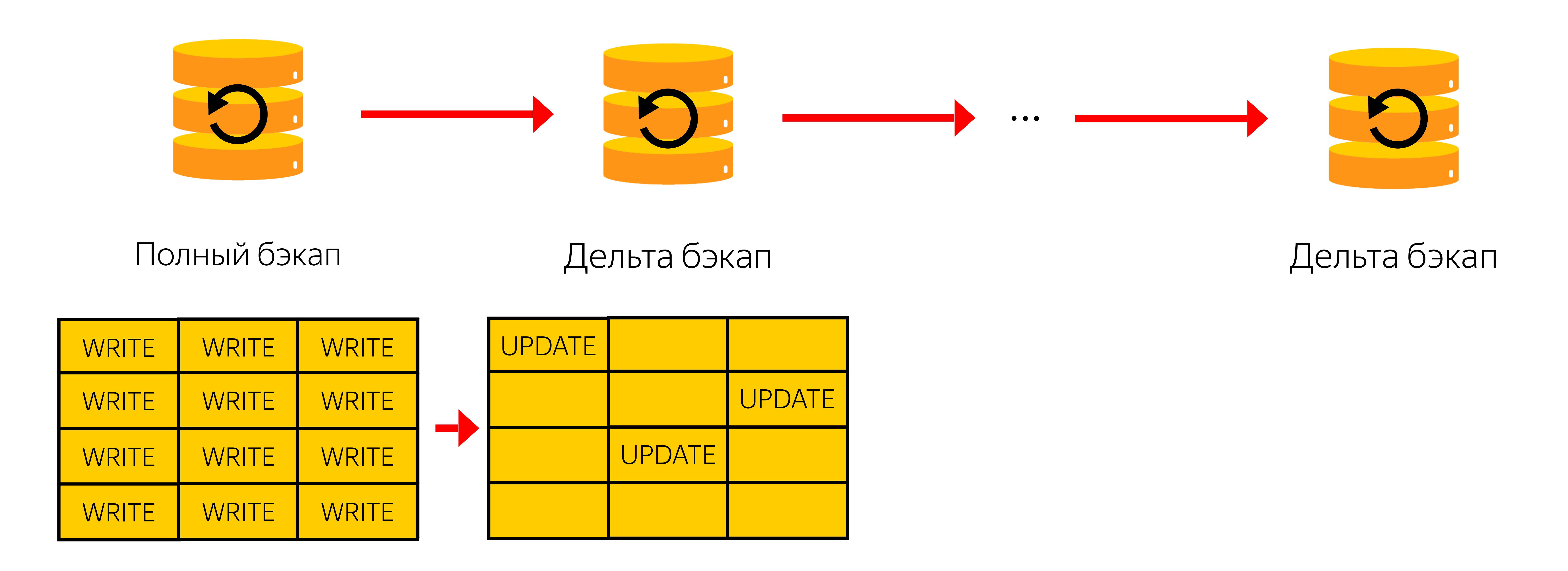


Порядок распаковки дельт

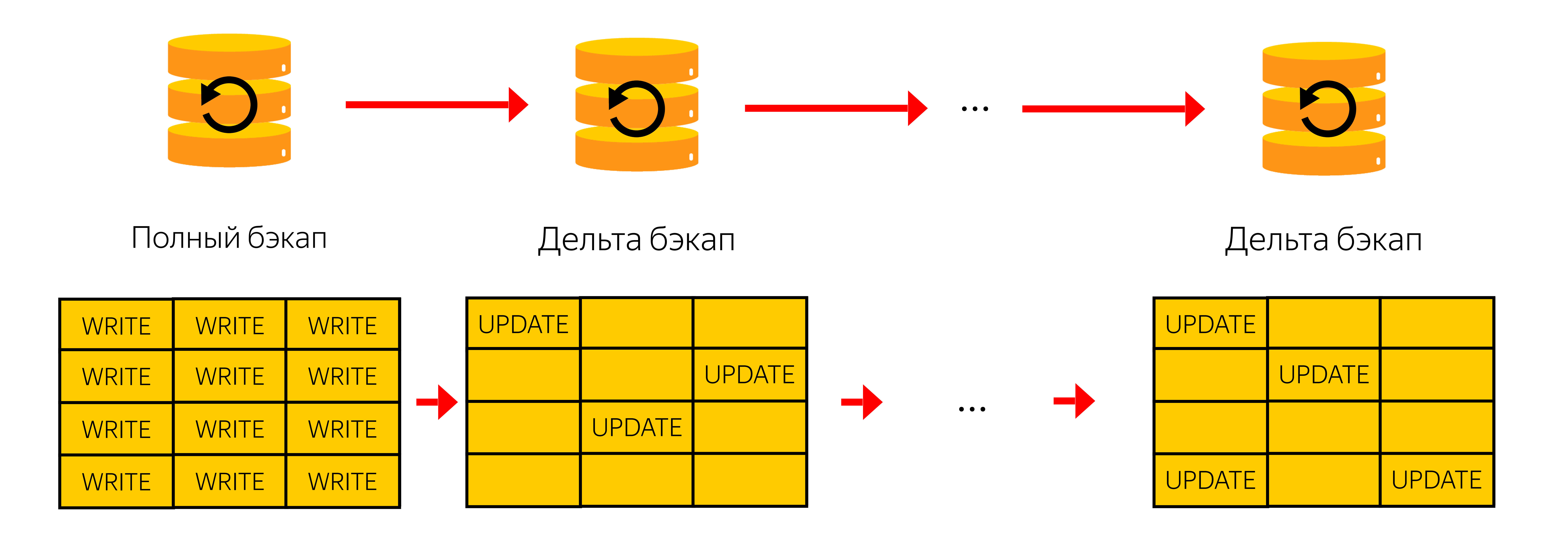


WRITE	WRITE	WRITE
WRITE	WRITE	WRITE
WRITE	WRITE	WRITE
WRITE	WRITE	WRITE

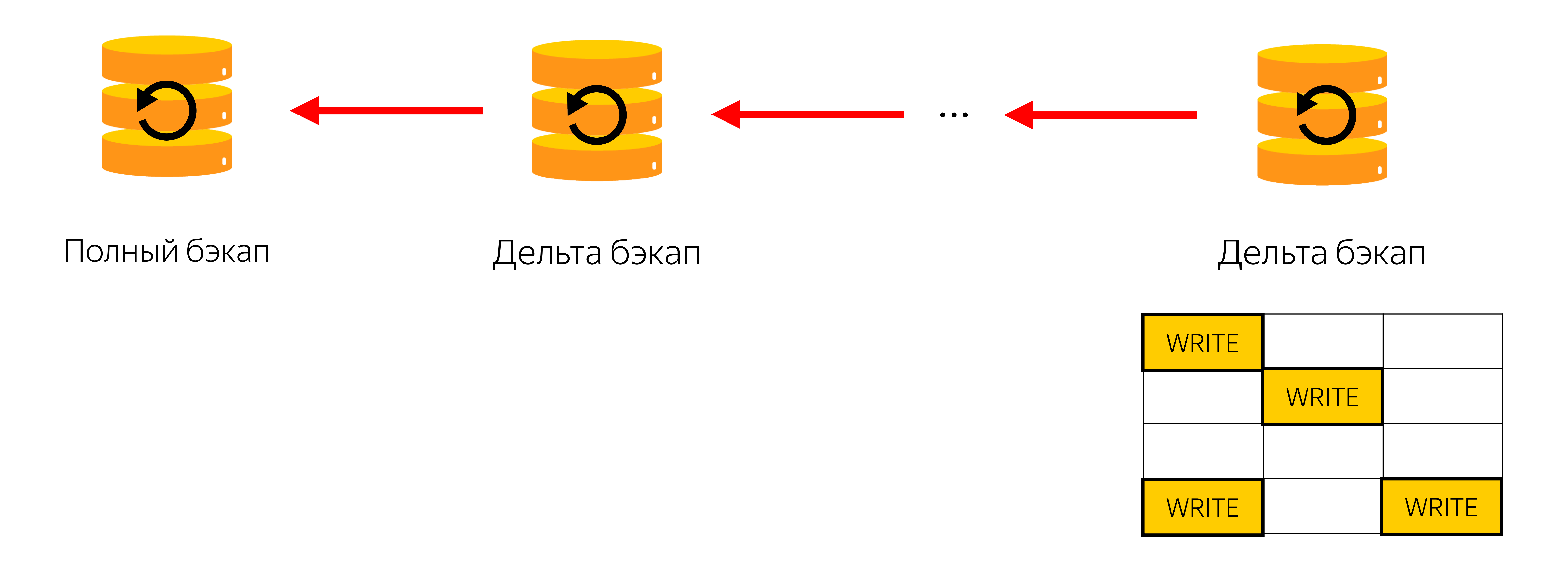
Порядок распаковки дельт



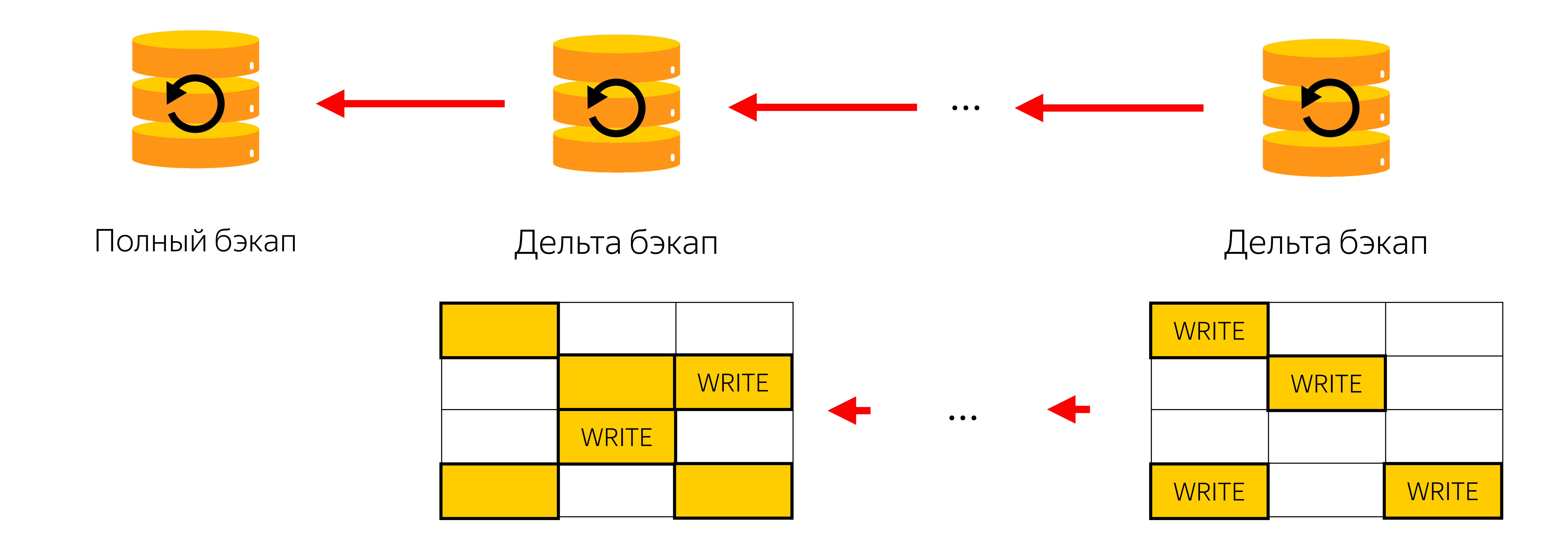
Порядок распаковки дельт



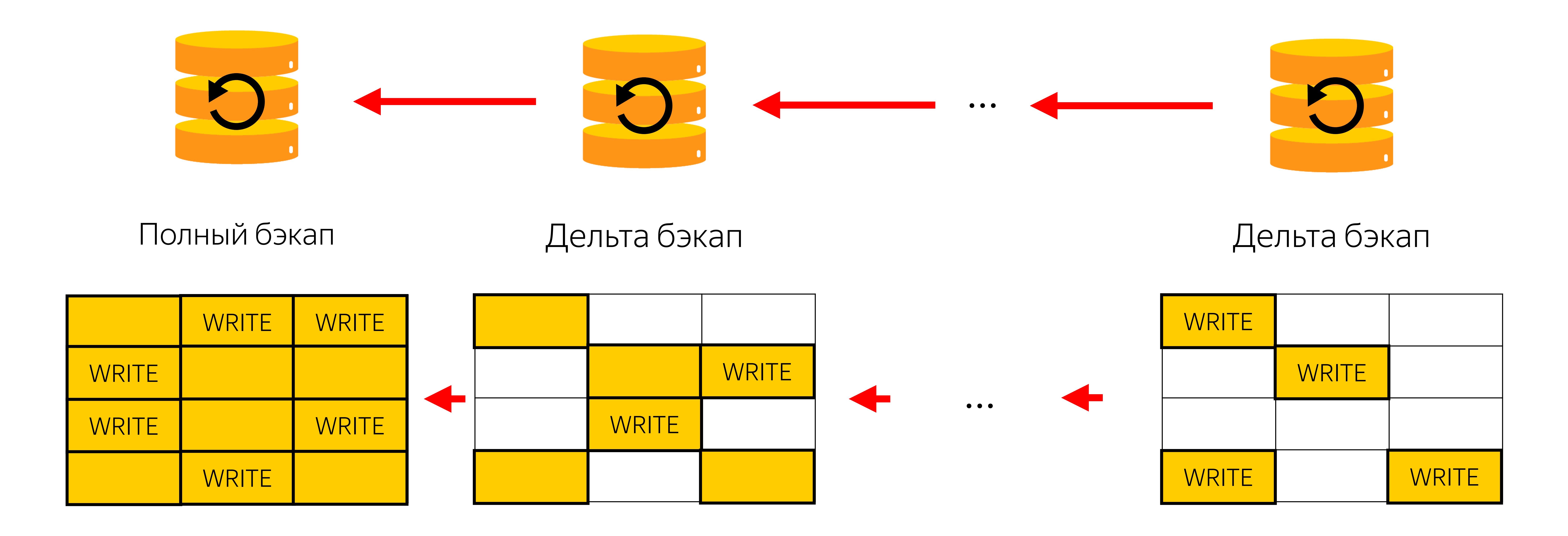
Обратный порядок распаковки дельт



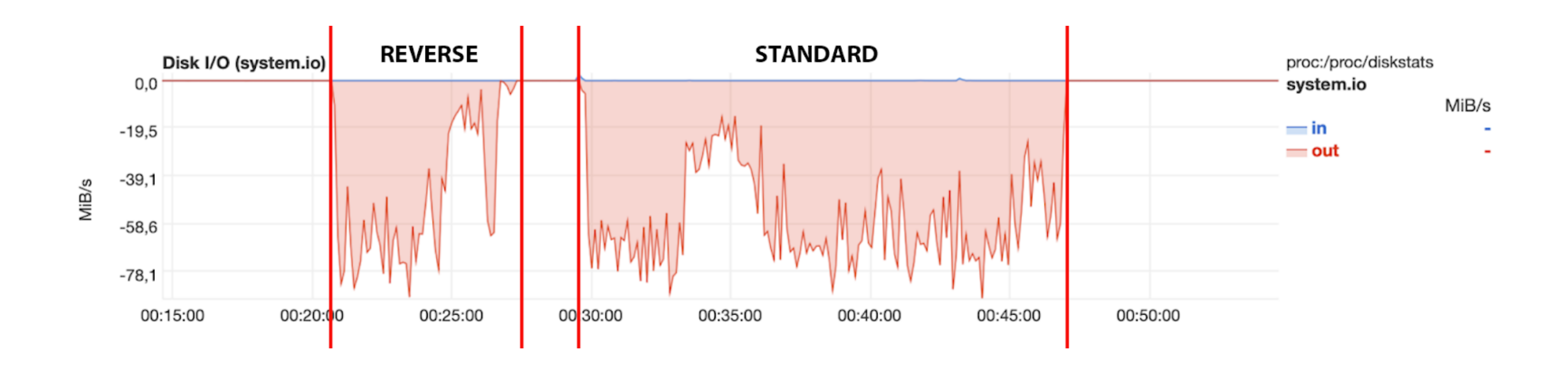
Обратный порядок распаковки дельт



Обратный порядок распаковки дельт



Насколько быстрее?



Как включить?

- С помощью флага:
  wal-g backup-fetch LATEST --reverse-unpack

  В конфиге:
  WALG_USE_REVERSE_UNPACK=TRUE

Бонус: пропуск загрузки ненужных архивов

- wal-g backup-push /path --rating-composer
- wal-g backup-fetch LATEST --reverse-unpack —skip-redundant-tars

Позволяет пропустить загрузку архивов в ходе обратной распаковки, в которых нет интересующих нас страниц

Как попробовать у себя?

Куда поставить звездочку?

github.com/wal-g/wal-g

Все новые фичи есть в последнем pre-release:

github.com/wal-g/wal-g/releases/tag/v0.2.22

Куда поставить звездочку?

github.com/wal-g/wal-g

Все новые фичи есть в последнем pre-release:

github.com/wal-g/wal-g/releases/tag/v0.2.22

А, вообще, у нас еще много идей...

Жду вопросов ☺

Даниил Захлыстов

Разработчик



